Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Niko Šuštar

Razvoj spletne aplikacije za načrtovanje relacijske podatkovne baze

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništva in informatike

Mentor: viš. pred. dr. Aljaž Zrnec

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Postopek načrtovanja je pri izdelavi kompleksnih sistemov nujen, saj omogoča razumeti in modelirati sisteme na različnih nivojih abstrakcije. Podobno je pri načrtovanju podatkovnih baz, kjer pa se pogosto zgodi, da se fazo konceptualnega načrtovanja enostavno preskoči. Rezultat omenjene faze je konceptualni model, ki prikazuje podatke in povezave med njimi, poleg tega pa je model primeren tudi za komunikacijo z naročnikom. V okviru diplomske naloge predstavite postopek načrtovanja podatkovne baze in izpostavite pomen konceptualnega načrtovanja. Razvijte spletno aplikacijo, ki bo na enostaven način omogočila izdelavo podatkovnega modela v vseh fazah načrtovanja podatkovne baze.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Niko Šuštar sem avtor diplomskega dela z naslovom:

Razvoj spletne aplikacije za načrtovanje relacijske podatkovne baze (angl. *Development of web application for designing a relational database*)

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Aljaža Zrneca;
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela;
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 17. oktobra 2016 Podpis avtorja:

Zahvaljujem se mentorju viš. pred. dr. Aljažu Zrnecu za pomoč, nasvete in usmerjanje do zaključka diplomskega dela. Prav tako se zahvaljujem tudi moji družini, prijateljem in vsem, ki so mi pomagali do zaključka študija.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
SQL	structured query language	sestavljen jezik za poizvedbe
DFD	data flow diagram	diagram toka podatkov
ERM	entity relationship modeling	model entiteta–razmerje
SUPB	database management system	sistem za upravljanje s podatkovnimi zbirkami
URL	Uniform Resource Locator	enolični krajevnik vira

Povzetek

Naslov: Razvoj spletne aplikacije za načrtovanje relacijske podatkovne baze

V diplomskem delu bom predstavil in prikazal, kako sem naredili spletno aplikacijo za načrtovanje relacijske podatkovne baze. Namen spletne aplikacije je načrtovalcem podatkovnih baz v nasprotju z ostalimi plačljivimi aplikacijami, ki za osnovo uporabljajo logični nivo, omogočiti brezplačno načrtovanje v konceptualnem nivoju.

Na začetku diplomskega dela na kratko predstavim nivoje načrtovanja podatkovne baze. V nadaljevanju razložim, kakšno dodano vrednost ima naš izdelek, ki za osnovo uporablja konceptualni nivo, v primerjavi z ostalimi. Po umestitvi izdelka v nivo načrtovanja umestim in predstavim še način načrtovanja same aplikacije. V drugem bolj praktičnem delu predstavim vsa uporabljena orodja in knjižnice, ki sem jih uporabil, in v zadnjem delu opišem in predstavim vnosne maske ter glavne funkcionalnosti izdelka.

Ključne besede: razvoj spletne aplikacije za načrtovanje relacijske podatkovne baze, podatkovna baza, konceptualni model, entitete, podatkovne relacije.

Abstract

Title: Development of web application for designing a relational database

In the graduate thesis I will present and show how I made a web application for designing relational databases. The purpose of the web application is to enable a database designer as opposed to the rest of paid applications that use logical level, free design in the conceptual level.

At the beginning of the thesis I briefly introduce the planning phases of the database. Below I explain what added value has our product using conceptual level in comparison with the others. After placement of the product in the level of planning, I present the way of planning within the application itself. In the second more practical part are described all the used tools and libraries we have used, and in the last part I describe and introduce the input mask and the main functionality of the product.

Keywords: development of web application for designing a relational database, data base, conceptual model, entity, data relations.

Poglavje 1

Uvod

Dandanes rastejo potrebe po novih in boljših aplikacijah. Ker veliko aplikacij uporablja za shranjevanje podatkov podatkovne baze, bi morala rasti tudi potreba po načrtovanju le-teh. V zadnjem času je zelo moderno agilno vodenje projektov in ker manjša programerska podjetja nimajo denarja za nakup dragih načrtovalskih orodij, se načrtovanju in dokumentiranju podatkovnih baz velikokrat izognejo, še zlasti fazi izdelave konceptualnega modela. Z razlogom, da bi čim več programerjev navdušili nad načrtovanjem in dokumentiranjem, kar bi olajšalo spremembe v podatkovnem modelu in zmanjšalo možnost napak, sem se odločil, da bom razvili aplikacijo, ki bo omogočila prav to in bo dostopna preko spleta.

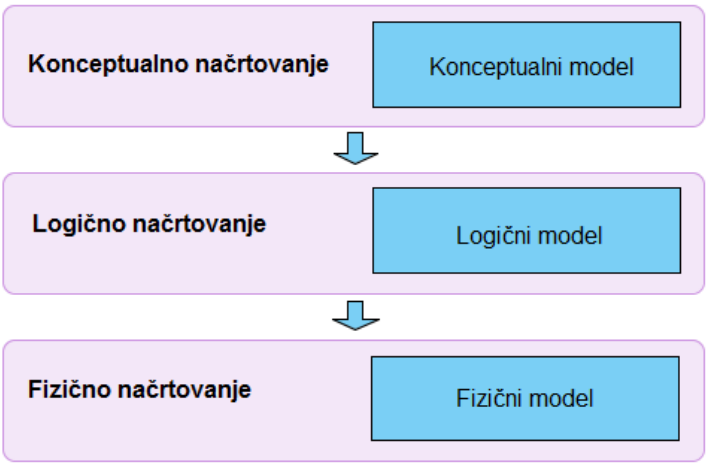
V diplomskem delu je predstavljen celoten proces načrtovanja podatkovne baze, kjer sem se še posebej osredotočil na fazo konceptualnega načrtovanja. V nadaljevanju je predstavljena spletna aplikacija, narejena v okolju ASP.NET MVC, katera za shranjevanje podatkov uporablja SQL podatkovno bazo. Aplikacija je zasnovana tako, da nam omogoča izdelati konceptualni model, ki se ga izdela v fazi konceptualnega načrtovanja, ko analitik intenzivno komunicira z naročnikom in mora biti uporabljena modelirna tehnika čim bolj enostavna za uporabo. Aplikacija nam poleg izdelave konceptualnega modela omogoča pretvorbo konceptualnega modela v logičen model. Narejena je tako, da nam na zelo pregleden način prikaže podatkovni model, zaradi

česar lahko veliko hitreje preverimo, ali je model dovolj dobro normaliziran in ali ustreza vsem omejitvam integritete. Poleg naštetega nam aplikacija omogoča pretvorbo v fizični model (naredi SQL stavke za kreiranje tabel, indeksov itd.) in izdelavo PDF dokumenta z opisom vseh tabel in atributov, ki nam kasneje lahko služi kot dokumentacija. V zaključku so podane še sklepne ugotovitve in podane možnosti za nadaljnji razvoj.

Poglavje 2

Načrtovanje podatkovne baze

Namen načrtovanja podatkovne baze je izdelati kakovosten načrt, na podlagi katerega bo mogoče kreirati podatkovno bazo, ki bo omogočala na urejen način shranjevati podatke in bo ustrezala vsem uporabniškim zahtevam. Načrtovanje je razdeljeno na tri povezane nivoje, kot je prikazano na sliki 2.1. V okviru vsakega nivoja nastane podatkovni model, ki podrobneje opisuje podatke, predstavljene s predhodnim podatkovnim modelom. Vsak model (na sliki 2.1 moder pravokotnik) prikazuje podatke na določenem nivoju abstrakcije in služi potrebam razvoja in komunikaciji z ustreznimi deležniki.



Slika 2.1: Nivoji načrtovanja podatkovne baze.

2.1 Konceptualno načrtovanje

2.1.1 Predstavitev

Prvi nivo načrtovanja podatkovne baze je konceptualno načrtovanje. S pomočjo konceptualnega načrtovanja opredelimo podatkovne potrebe in poskrbimo za opis pomena podatkov za poslovno domeno. Oblikovanje ali načrtovanje konceptualnega modela se začne s temeljitim pregledom obstoječe dokumentacije. Tukaj je zelo pomembno sodelovanje in komunikacija z uporabniki, ki so nosilci znanja o poslovni domeni. Hkrati pa moramo upoštevati vsa poslovna pravila. Takšnega načrtovanja ne moremo avtomatizirati. Za konceptualni nivo načrtovanja so običajno zadolženi in odgovorni analitiki. Ker pa se napake prenašajo naprej na naslednje modele, je v tem delu načrtovanje najbolj kritično [3, str. 45–49].

Konceptualni model bi moral izpolnjevati naslednje cilje:

- omogočiti enostavno razumevanje podatkovnih potreb in povezav med podatki;
- omogočiti enostavno komunikacijo med analitikom in naročnikom in
- služiti kot dokumentacija za kasnejšo uporabo.

2.1.2 Tehnike za predstavitev podatkovnih modelov

Obstaja več različnih tehnik, s katerimi lahko načrtujemo in predstavimo konceptualni model. Ker izmed vseh spodaj naštetih tehnik v okviru diplome razvita aplikacija temelji samo na tehniki, ki za predstavitev modela uporablja entitetno-relacijske diagrame, bom predstavil le to tehniko.

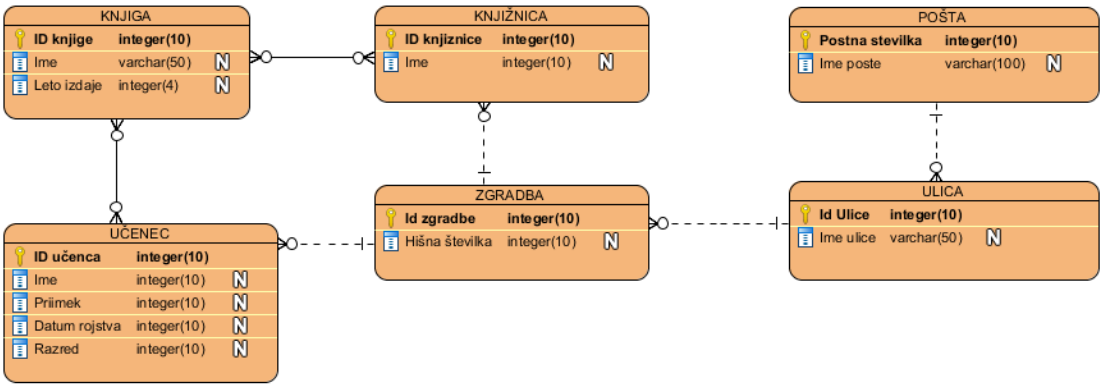
Za predstavitev konceptualnega modela se uporablja [20]:

- entitetno-relacijske diagrame (ER diagram);
- razširjene diagrame entiteta-razmerje (EER diagram);

- diagrame razredov (UML notacije) in
- diagrame model-vloga (object-role model).

ER diagram (ang. Entity relationship diagram)

Zelo znana tehnika za načrtovanje in prikaz konceptnega modela je načrtovanje z ER diagrami. Namen ER diagramov je opisati in opredeliti poslovno domeno poslovnih podatkov ter prikazati razmerja med podatki. S to diagramsko tehniko vizualiziramo poslovne podatke in razmerja med njimi, kot je to prikazano na Sliki 2.2.



Slika 2.2: Primer modela entiteta-razmerje.

Gradniki entitetno-relacijskih diagramov

Entitetni tip - Entitetni tip združuje objekte (entitete) poslovne domene z istimi lastnostmi. Na Sliki 2.2 so to imena, ki so napisana na vrhu vsakega oblačka. Vsak entitetni tip je neodvisen od ostalih in ga je mogoče enolično identificirati. Za vsak entitetni tip moramo poznati ime oziroma vedeti, kaj predstavljajo entitete, ki mu pripadajo [1, str. 23–28]. Entitetni tip je lahko fizični predmet, na primer avto, hiša, oseba itd., lahko pa je tudi dogodek, npr. naročilo, rezervacija itd. V žargonu se velikokrat namesto izraza entitetni tip uporablja izraz entiteta.

Atribut - Vsak entitetni tip ima določene lastnosti, kot so na primer ime, barva, velikost, število itd. Na Sliki 2.2 so to besede znotraj pravokotnikov. Tem lastnostim v entitetno-relacijskem diagramu pravimo atribut. Vsaka lastnost je svoj atribut. Entitetnemu tipu pa določimo le tiste lastnosti, ki so nam pomembne za poslovno domeno. Na koncu vsakemu atributu posebej določimo podatkovni tip (numerični tip, znakovni ...), dolžino oziroma velikost podatka in pa ali bo podatek obvezen ali ne.

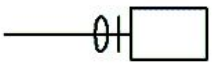
Enolični identifikator entitete - Enolični identifikator entitete je sestavljen iz enega ali več atributov entitetnega tipa [5, str. 64–65]. Vsak identifikator določa natanko eno entiteto v množici entitet (entitete, ki pripadajo istemu entitetnemu tipu). Entitetni indentifikator pogosto imenujemo tudi ključ. Na Sliki 2.2 so to vsi atributi, ki imajo pred imenom narisan rumen ključ.

Razmerje - Razmerja si lahko predstavljamo kot glagole, ki povezujejo samostalnike. Na primer: človek je lastnik avta. V tem primeru razmerje 'je lastnik' poveže dva entitetna tipa, in sicer entitetni tip 'človek' in entitetni tip 'avto'. S tem smo določili odnos med dvema entitetama. Vsakemu razmerju pa je pomembno določiti še številnost in obveznost. Obveznost pove, ali morata biti dve entiteti obvezno v razmerju, ali lahko tudi nista. S številnostjo pa določimo, koliko entitet enega tipa je lahko povezano z entiteto drugega tipa [5, str. 62–63].

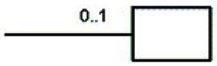
Številnosti razmerja:

- Številnost: nič ali ena. Če imamo povezavo med entitetnim tipom A in entitetnim tipom B, kjer je razmerje A proti B enako nič ali ena, potem s številnostjo 'nič ali ena' povemo, da bo lahko imela vsaka entiteta v entitetnem tipu A povezavo na entiteto v entitetnem tipu B. Primer številnosti: Dirkalni avto ima voznika. Obstajajo pa avti, ki voznika še nimajo oziroma nimajo več. V entitetno-relacijskem diagramu lahko

številnost označimo kot je prikazano na Sliki 2.3, lahko pa tudi tako, kot je prikazano na Sliki 2.4.

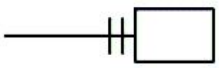


Slika 2.3: Grafična notacija za številnost 0 ali 1 [10].

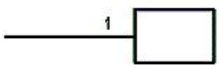


Slika 2.4: Številska notacija za številnost 0 ali 1 [10].

- Številnost natanko ena. Če imamo povezavo md entitetnim tipom A in entitetnim tipom B, kjer je razmerje A proti B enako natančno ena, potem je vsaka entiteta iz A povezana na natančno eno entiteto iz entitetnega tipa B. Primer številnosti: Vsako stanovanje ima natančno en naslov. Primer označbe je prikazan na Sliki 2.5 in na Sliki 2.6.



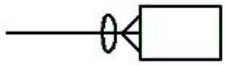
Slika 2.5: Grafična notacija za številnost natanko ena [10].



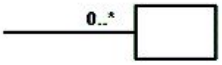
Slika 2.6: Številska notacija za številnost natanko 1 [10].

- Številnost: nič ali več. Če imamo povezana entitetna tipa A in B, kjer je razmerje od A proti B definirano s številnostjo 'nič ali več', potem

ima lahko vsaka entiteta v entitetnem tipu A več povezav na entitete iz entitetnega tipa B. Lahko pa nima nobene. Številnosti sta prikazani na Sliki 2.7 in na sliki 2.8. Primer številnosti: Vsak profesor ima nič ali več asistentov.

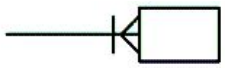


Slika 2.7: Grafična notacija za številnost 0 ali N [10].

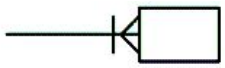


Slika 2.8: Številska notacija za številnost 0 ali N [10].

- Številnost: ena ali več. Če imamo povezana entitetna tipa A in B, kjer je razmerje od A proti B definirano s številnostjo 'ena ali več', potem ima vsaka entiteta v entitetnem tipu A vsaj eno povezavo na entitete iz entitetnega tipa B. Načina označbe sta prikazana na Sliki 2.8 in na Sliki 2.9. Primer številnosti: Vsaka hrana ima eno ali več sestavin.



Slika 2.9: Grafična notacija za številnost 1 ali N [10].



Slika 2.10: Številska notacija za številnost 1 ali N [10].

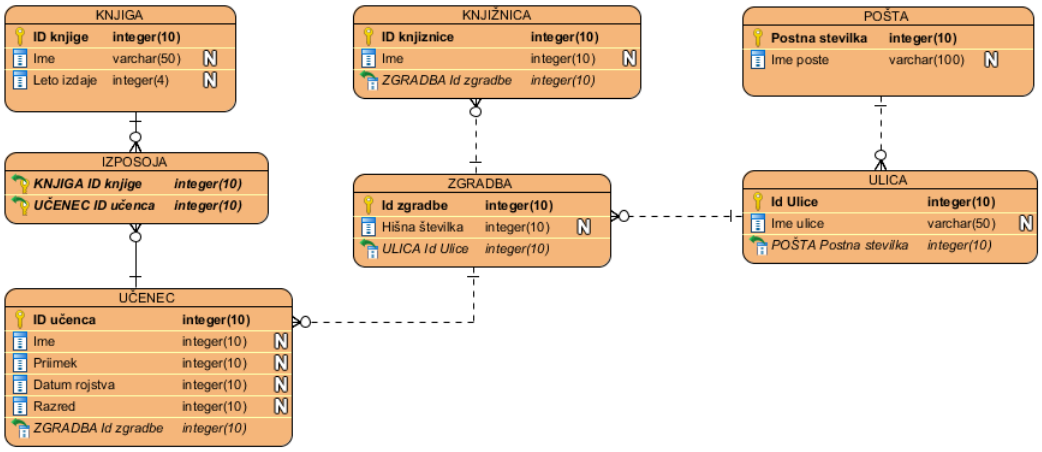
Analiza (konceptualno načrtovanje)	**Načrtovanje (logično načrtovanje)**
Konceptualni model	Relacijski model
Entitetni tip	Relacija
Entiteta	N-terica
Atribut	Atribut
Enolični identifikator	Ključ
Povezava 1:n	Tuji ključ
Povezava m:n	Vmesna tabela

Tabela 2.1: Pretvorba elementov konceptualnega modela v elemente relacijskega modela.

2.2 Logično načrtovanje

Drugi nivo načrtovanja predstavlja logično načrtovanje. Namen logičnega načrtovanja je izdelati logičen model podatkovne baze, ki prikazuje elemente logičnega podatkovnega modela, v našem primeru so to relacije z atributi in ključi. V okviru predstavljenega procesa načrtovanja se logični model izdela na podlagi konceptualnega modela z uporabo transformacijskih pravil [3, str. 479]. Tabela 2.1 prikazuje primer pretvorbe elementov konceptualnega modela v relacijski podatkovni model.

V prehodu iz konceptualnega modela v logični model se močni in šibki entitetni tipi enolično preslikajo v relacije. Nato se vsem relacijam določi primarne ključe. Primarni ključi so identifikatorji, ki določajo točno eno n-terico (zapis) znotraj relacije. Za primarne ključe vzamemo enake atribute, katere smo v konceptualnem nivoju označili kot enolične identifikatorje. Po preslikavi vseh entitetnih tipov v relacije, preslikamo povezave med njimi. Pri večštevnih relacijah (0:n in 1:n) dodamo tuje ključe. V relacijah, kjer sta obe razmerji v povezavi večštevni (povezave z relacijo 'm:n' oziroma povezavo, kjer imata obe relaciji v razmerju, številnost eden proti drugim enako 'nič ali več' ali pa 'ena ali več'), naredimo vmesno relacijo, ki vsebuje primarna ključa obeh relacij. Prehod iz konceptualnega modela v logični model je navadno



Slika 2.11: Primer logičnega modela.

avtomatiziran v CASE orodjih. Na Sliki 2.11 je logični model, ki je nastal iz konceptualnega modela, ki je prikazan na Sliki 2.2.

2.3 Fizično načrtovanje

Zadnji nivo načrtovanja podatkovne baze predstavlja fizično načrtovanje. V tem nivoju predstavimo podatkovni model, kjer upoštevamo zmogljivost in omejitve našega končnega sistema za upravljanje s podatki. Končni fizični podatkovni model je predstavljen z množico SQL stavkov za kreiranje različnih objektov v bazi (tabele, atributi, indeksi itd.). Primer fizičnega modela je prikazan na Sliki 2.12.

Glavne naloge fizičnega načrtovanja so ustvariti vse sestavljene oziroma izračunljive atribute (atribute, ki se izračunajo s sprožilci (ang. triger), npr. starost), določiti fizično strukturo podatkovne baze, identificirati različne skupine uporabnikov in z denormalizacijo ter indeksiranjem izboljšati učinkovitost sistema.

```
CREATE TABLE Knjiga (
    IdKnjige Int NOT NULL,
    Ime NVarChar(50) NOT NULL,
    LetoIzdaje Int NOT NULL,
    CONSTRAINT [PK_Knjiga] PRIMARY KEY CLUSTERED ([IdKnjige] ASC) ON[PRIMARY]
)

GO

ALTER TABLE Knjiga WITH CHECK ADD  CONSTRAINT [FK_Knjiga_Izposoja] FOREIGN KEY(IdKnjige)
REFERENCES Izposoja (IdKnjige)
GO

ALTER TABLE Knjiga CHECK CONSTRAINT [FK_Knjiga_Izposoja]
GO


CREATE TABLE Izposoja (
    Id Int NOT NULL,
    IdKnjige Int NOT NULL,
    IdUcenca Int NOT NULL,
    CONSTRAINT [PK_Izposoja] PRIMARY KEY CLUSTERED ([Id] ASC) ON[PRIMARY]
)

GO
```

Slika 2.12: Primer fizičnega modela.

2.4 Načrtovanje po korakih

Kot sem že prej omenil, je konceptualno načrtovanje najbolj kritično za razvijalce programske opreme. Predno začnemo razmišljati, katere podatke bi shranjevali v podatkovno bazo, moramo opredeliti problemsko domeno poslovnih podatkov in poznati vse poslovne procese, ki bodo na kakršen koli način komunicirali s podatkovno bazo. Na ta način se znebimo potencialnih napak, ki se ne prenesejo samo v naslednji nivo načrtovanja, ampak tudi v končno podatkovno bazo in posledično v aplikacijo.

V nadaljevanju so najprej našteti vsi koraki načrtovanja, ki so razdeljeni po fazah, nato pa je vsak korak še podrobneje opisan.

Koraki konceptualnega načrtovanja:

1. identifikacija entitetnih tipov;
2. identifikacija atributov;
3. identifikacija razmerij;
4. izbira kandidatov za identifikatorje in izbira primarnega identifikatora;

5. validiranje konceptualnega modela;
6. preverjanje konceptualnega modela z uporabnikom.

Koraki logičnega načrtovanja:

1. pretvorba konceptualnega modela v logični model;
2. normalizacija (zmanjševanje ponavljanja podatkov);
3. preverjanje omejitve integritete.

Koraki fizičnega načrtovanja:

1. pretvorba logičnega modela v fizični model ciljnega SUPB;
2. indeksiranje (izboljševanje učinkovitosti);
3. izdelava načrta uporabniških pogledov;
4. načrtovanje varnostnih mehanizmov;
5. denormalizacija (izboljševanje učikovitosti).

2.4.1 Identifikacija entitetnih tipov

Entitetni tip združuje objekte (entitete) poslovne domene z istimi lastnostmi. Namen načrtovanja entitetnih tipov je, da ločimo vse nepovezane koncepte, oziroma da vsak entitetni tip predstavlja samo en koncept [4, str. 56]. Na primer: ker 'uporabnik' in 'naročilo' predstavljata čisto drugačni stvari, bi morala biti to dva različna entitetna tipa.

Če smo si prej naredili diagram toka podatkov, potem lahko uporabimo imena entitet in objektov za shranjevanje podatkov iz diagrama za naše entitetne tipe. V nasprotnem primeru pa imamo na voljo več tehnik za identificiranje entitetnih tipov. Ena izmed tehnik je pregled uporabniških zahtev. S slednjo pregledamo vse omenjene samostalnike in fraze (npr. hitrost, izdelava, avto, voznik, datum ...). Nato poskušamo ločiti vse objekte (npr.

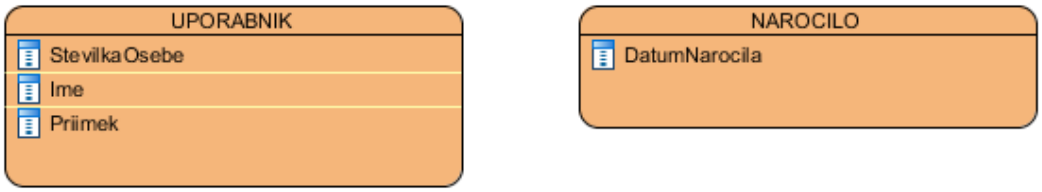
avto in voznik) od lastnosti objektov (hitrost, izdelava ...). Entitete z istimi lastnostmi predstavljajo primerke enega entitetnega tipa. Lastnosti, ki jih pripišemo entitetam, postanejo njeni atributi. Težava v načrtovanju se pojavi, ker entitete niso jasno razvidne iz dokumentacije in ker je včasih težko ločiti, kaj je atribut in kaj entiteta. Ker je načrtovanje subjektivne narave in za določeno problemsko domeno lahko dobimo več različnih rešitev, je končni entitetno-relacijski diagram zelo odvisen od izkušenj in sposobnosti načrtovalca. Na Sliki 2.13 sta prikazana dva entitetna tipa, ki prikazujeta primer konceptualnega modela po končanem koraku identifikacije entitetnih tipov.

Slika 2.13: Primer konceptualnega modela po identifikaciji entitetnih tipov.

2.4.2 Identifikacija atributov

Z načrtovanjem atributov identificiramo lastnosti entitetnih tipov, kot je prikazano na Sliki 2.14. Vsakemu entitetnemu tipu dodamo enega ali več atributov, odvisno od tega, katere podatke hočemo v bazi hraniti. Načrtovanje lahko poteka tako, da s tehniko preučevanja uporabniških zahtev iščemo samostalnike, ki predstavljajo lastnosti. Ko imamo zbrane vse lastnosti, jih po pomenu dodamo v entitetne tipe. Ko smo dodali vse atribute, moramo preveriti, ali so vsi atributi atomarni, ali imamo prisotne tudi sestavljene in večvrednostne atribute.

Ob dodajanju atributov v entitetne tipe je smiselno, da so poimenovani tako, da v čim bolj človeku prijazni obliki opišejo polje. Na primer, ko imamo atribut, ki predstavlja številko osebe, ga ne poimenujemo 'Atribut1', ampak 'StevilkaOsebe'. Poleg slednjega pravila se pri poimenovanju izogibamo presledkov, šumnikov in ostalih znakov, ki niso črke. Veliko podjetij pa ima

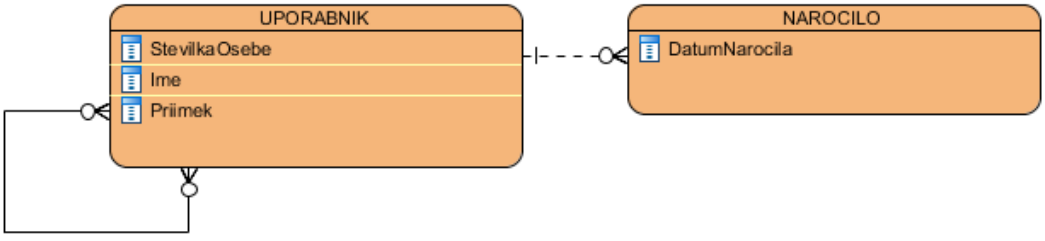


Slika 2.14: Primer konceptualnega modela po identifikaciji atributov.

svoje standarde in smernice za poimenovanje entitetnih tipov in atributov, npr. prve tri črke atributa opisujejo atributov entitetni tip.

2.4.3 Identifikacija razmerij

V realnem svetu se zelo malokrat zgodi, da imamo v modelu samo en entitetni tip ali pa da imamo čisto nepovezane entitetne tipe. Za identificiranje razmerij oziroma povezav med entitetnimi tipi lahko zopet uporabimo pristop, kjer pregledamo uporabniške zahteve. Razlika med prejšnjima pregledoma je v tem, da sedaj iščemo glagole, ki kakor koli povezujejo naše entitetne tipe (npr. uporabnik 'je oddal' naročilo, skakalec 'je skočil' rekordno dolžino skakalnice, študent 'si izbere' mentorja). V modelu kreiramo le tiste povezave, ki predstavljajo razmerja med podatki, ki so za nas pomembna. Pozorni moramo biti tudi na tiste povezave, ki povezujejo več entitetnih tipov ali pa povezujejo entitetni tip s samim sabo [4, str. 58].



Slika 2.15: Primer konceptualnega modela po indentifikaciji relacij.

Ko najdemo povezavo, moramo tej določiti številnost. To naredimo tako, da določimo minimalno in maksimalno število povezav entitete na drug entitetni tip. Za določanje minimuma se vprašamo, "ali moramo imeti"; če je odgovor 'DA', potem je minimum število ena, v nasprotnem primeru pa je minimum število nič. Za določanje maksimalnega števila pa se vprašamo "koliko". Ker se izogibamo določitvi točne maksimalne vrednosti, za odgovor z vrednostjo večjo od ena uporabimo izraz 'več' ali pa matematični izraz za cela števila 'n'.

2.4.4 Izbira kandidatov za identifikatorje in izbira primarnega identifikatora

Vsakemu entitetnemu tipu moramo določiti primarni identifikator. Za določitev ključev obstajata dve strategiji.

Pri prvi strategiji za vsak entitetni tip določimo kandidate za identifikatorje. Pri izbiranju potencialnih identifikatorjev moramo vedeti, da bo atribut moral vsebovati samo unikatne vrednosti, zato atributi, ki vsebujejo imena ali nazive oziroma se lahko ponavljajo, niso primerni. Druga stvar, na katero moramo biti pozorni, je tip atributa. Če imamo na voljo več kandidatov, vedno izberemo atribut, ki ima celoštevilsko vrednost ali pa ima najmanjšo dolžino znakov. S tem pa vplivamo na hitrejšo izvedbo bodočih poizvedb.

V drugi strategiji enostavno naredimo nov atribut, ki ga nekateri imenujejo nadomestni identifikator. Ta atribut nima nobene povezave s problemsko domeno, ampak bo služil izključno samo kot primarni identifikator, kasneje kot ključ.

2.4.5 Validiranje konceptualnega modela

Kljub temu, da je korak načrtovanja zelo zamuden, je zelo pomemben. Z validacijo preverimo, če model vsebuje odvečne elemente in če zdrži vse transakcije, ki bile zahtevane.

Preverjanje obstoja odvečnih elementov začnemo s pregledom vseh ena proti ena povezav. Če obstajajo entitetni tipi, ki imajo enake atribute oziroma predstavljajo enake objekte, jih je potrebno združiti. V primeru, da imamo entitetni tip 'Zaposlen' in entitetni tip 'Študent', ki imata enake lastnosti, potem entitetna tipa združimo v entitetni tip 'Delavec'. Če bi med združenimi entitetnimi tipi radi razlikovali, potem dodamo nov atribut. Za prejšnji primer bi lahko poimenovali atribut 'TipOsebe'. Po preverjanju odvečnih entitetnih tipov preverimo odvečne povezave. Povezava je odvečna, če je do enake informacije možno priti z uporabo drugih povezav. Vseeno pa je potrebno paziti, ker imajo lahko povezave različen pomen.

V drugem delu validacije preverimo, ali model zdrži vse transakcije, ki se bodo izvajale na podatkovni bazi. Preveriti moramo vse zahtevane vnose, urejanja, brisanja in poizvedbe. Za preverjanje transakcijskih zahtev se uporabljata dva pristopa – preverjanje opisa transakcij in preverjanje transakcijskih poti. Z validacijo lahko odkrijemo, ali se določena transakcija lahko izvede, posebej obremenjene dele podatkovnega modela in dele modela (entitetni tipi), ki se sploh ne uporabijo.

2.4.6 Preverjanje konceptualnega modela z uporabnikom

Pri predstavitvi modela uporabniku preverimo, če model ustreza vsem uporabniškim zahtevam. Za lažjo predstavo naročniku oziroma uporabniku pripravimo prototipe najpomembnejših transakcij, s katerimi dodatno preverimo narejeni model. Z uporabnikom torej preverimo, če smo s pregledom specifikacije oziroma pregledom virov našli vse entitetne tipe, pravilno razbrali vsa razmerja med relacijami, določili vse atribute in jim priredili pravilne podatkovne tipe.

2.4.7 Pretvorba konceptualnega modela v logični model

Razvit konceptualni model je izhodišče za izdelavo logičnega modela. V pretvorbi konceptualnega modela v logični model je potrebno tega pretvoriti v obliko, ki je razumljiva našemu ciljnemu sistemu za upravljanje s podatkovnimi zbirkami.

Vse entitetne tipe preslikamo v relacije. Primarni identifikatorji postanejo primarni ključi relacij. Pri večštevnih povezavah (0:n in 1:n) dodamo tuje ključe. Namesto povezav s številnostjo mnogo proti mnogo dodamo vmesno (povezovalno) relacijo, kjer sta tuja ključa primarna ključa relacije, ki ju povezuje [4, str. 103–105].

2.4.8 Normalizacija

Normalizacija je postopek, ki se v okviru tega pristopa načrtovanja uporablja kot tehnika za preverjanje, ali se dobljene relacije nahajajo v želeni normalni obliki. Z normalizacijo dosežemo, da se podatki v bodoči podatkovni bazi ne bodo ponavljali. Na primer, če imamo v entitetnem tipu 'Oseba' atribut naslov, se bo naslov lahko večkrat ponovil. Prva težava se pojavi, ker bi se lahko nekateri uporabniki zmotili pri vnosu, ali pa bi naslov enostavno skrajšali (npr. v bazi bi se pojavili zapisi 'Janezova ulica' in 'Janezova ul.'). Ko pa bi s poizvedbo hoteli videti, v katerem kraju živi največ uporabnikov, bi ugotovili, da poizvedba deluje napačno. Druga težava, ki se pojavi pri nenormaliziranih modelih, je težje ažuriranje podatkov (npr. če se spremeni naslov, moramo namesto enega zapisa spremeniti več zapisov in s tem prej naredimo kakšno napako) [2, str. 174].

2.4.9 Preverjanje omejitve integritete

V tem koraku za vsak atribut posebej preverimo obveznost podatka, določimo domeno in velikost atributa. Pri določanju domen moramo paziti, da podatkovna baza podpira izbran tip podatka.

2.4.10 Pretvorba logičnega modela v fizični model ciljnega SUPB

Za pretvorbo modela v ciljni SUPB moramo imeti izbran tip podatkovne baze (npr. Oracle, MS SQL itd.). Tip podatkovne baze pogosto definira že naročnik in je že naveden v specifikaciji. Če načrtujemo podatkovno bazo zase ali pa nam naročnik ponudi več možnih tipov, moramo tega izbrati sami. Eden izmed pogojev za izbiro tipa podatkovne baze je lahko operacijski sistem, ki ga uporablja server. Na primer SQL Server deluje samo na operacijskem sistemu Microsoft Windows, med tem ko na primer DB2 deluje na vseh UNIX operacijskih sistemih (Linux, Solaris ...), kot tudi na operacijskem sistemu Windows. Za izbiro podatkovne baze so pogojene tudi varnostne zahteve, velikost podatkovne baze, hitrost branja in pisanja, kot tudi potreba po ostalih zahtevah (npr. uporaba sprožilcev (ang. triger)).

Poleg izbire podatkovne baze moramo ob pretvorbi logičnega modela v ciljni SUPB spremeniti vse potrebne atribute (ang. column) v pravilne podatkovne tipe, kot tudi kreirati že prej definirane povezave. V tem koraku tudi določimo, kako bodo predstavljeni izpeljani atributi, ali bodo shranjeni v podatkovni bazi, ali se bodo izračunali ob vsaki poizvedbi.

2.4.11 Indeksiranje (izboljševanje učinkovitosti)

Indeks je kazalec na podatke v tabeli. Indeksi so shranjeni v posebnih podatkovnih strukturah (običajno B+ drevo), ki jih iskalni mehanizem podatkovne baze uporablja za hitrejše pridobivanje podatkov [19].

Z uporabo indeksov lahko pohitrimo iskanje podatkov, kot tudi povezovanje tabel, vendar upočasnimo spreminjanje in vnašanje novih podatkov.

2.4.12 Izdelava načrta uporabniških pogledov

Uporabniški pogled (ang. view) je navidezna tabela, ki fizično v podatkovni bazi ne obstaja, temveč se z vsakem poizvedovanjem kreira na novo. Z uporabniškim pogledom omogočimo pogled na podatke, ki jih določen uporabnik

potrebuje, in omejimo dostop in možnost spreminjanja podatkov (na tabeli in uporabniškem pogledu lahko nastavimo različne pravice).

2.4.13 Načrtovanje varnostnih mehanizmov

Namen varnostnih mehanizmov je omogočiti dostop samo omejenemu številu uporabnikov in zavarovati podatke. Navadno omejimo dostope z uvedbo uporabniških imen in gesel in z uvedbo različnih vlog uporabnikov. Na primer administratorju omogočimo neomejeno urejanje podatkovne baze, ostalim pa omogočimo samo branje in pisanje točno določenih podatkov.

2.4.14 Denormalizacija (izboljševanje učinkovitosti)

Normalizirane podatkovne sheme pogosto trpijo zaradi težav glede zmogljivosti. Z drugimi besedami, zmanjšali smo ponavljanje podatkov, nismo pa izboljšali zmogljivosti za dostop do podatkov, saj moramo namesto poizvedovanja po eni tabeli poizvedovati po več tabelah. Namen denormaliziranja pa je prav izboljšanje zmogljivosti. Denormalizacija navadno pohitri poizvedbe, ker nekatere atribute, ki smo jih ob normalizaciji prestavili v ločene tabele, vrnemo nazaj, vendar s tem upočasnimo spreminjanje podatkov, saj moramo namesto ene vrednosti spremeniti več njih [2, str. 178].

Poglavje 3

Razvoj spletne aplikacije

V poglavju bom predstavil razvoj spletne aplikacije. Razvoj spletne aplikacije zajema izbiro tehnologije, s katero sem naredil aplikacijo, načrtovanje spletne aplikacije in podatkovne baze ter vsa uporabljena orodja, ki sem jih uporabil za izgradnjo aplikacije.

Pred vsakim razvojem aplikacije si zastavimo cilj, katerega kasneje poskusimo realizirati. Za postavitev ciljev lahko uporabimo več tehnik. V mojem primeru je bila uporabljena tehnika 'brainstorming' z izhodiščno temo 'razvoj aplikacije za načrtovanje podatkovne baze'. V nadaljevanju so našteti zastavljeni cilji.

Zastavljeni cilji:

izdelati spletno aplikacijo, ki bo omogočala

- enostavno izdelavo konceptualnega modela;
- pretvorbo konceptualnega modela v logični model in
- pretvorbo logičnega modela v fizični model.

3.1 Izbira tehnologije

Razvijalci aplikacij se pogosto srečujejo s problemom izbire tehnologije za razvoj aplikacije. V mojem primeru je bil slednji problem še večji, ker se v praksi nisem še nikoli srečal z razvijanjem premikajočih objektov, še manj pa z njihovim grafičnim povezovanjem. Kljub prisotnem strahu pred neznanim, ki mi je dal še dodaten zagon, je bila želja narediti izdelek kot spletno aplikacijo. Dokončna odločitev, da bo aplikacija spletna aplikacija, je bila sprejeta s pomočjo spodnjih argumentov.

Prednosti spletnih aplikacij:

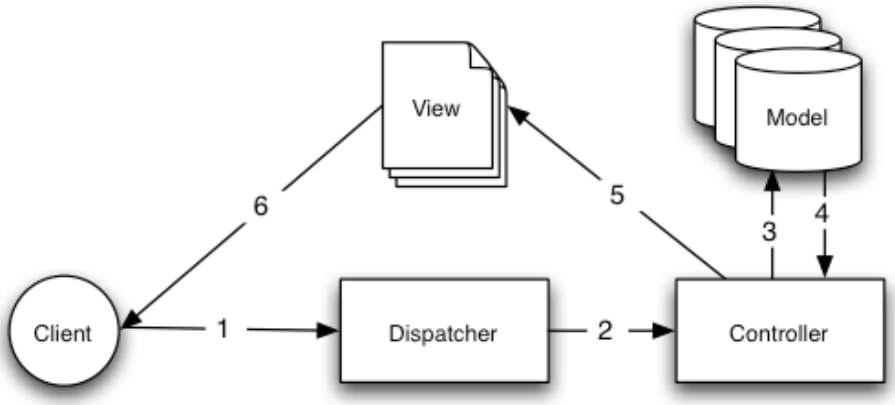
- Dostopnost: Za dostop ni potrebno namestiti nobene namenske programske opreme, ki uporabnika še dodatno odvrne od uporabe.
- Kompatibilnost: Spletne aplikacije so preko spletnih brskalnikov dostopne iz najrazličnejših operacijskih sistemov.
- Lažja nadgradnja: Če želimo spremeniti vsebino ali pa samo odpraviti napako, bodo spremembe takoj vidne vsem uporabnikom. Pri običajnih aplikacijah pa moramo uporabnika prisiliti, da si prenese posodobitev. Pogoste posodobitve pa so lahko zelo nadležne, kar lahko privede do izgube uporabnikov.
- Lažja promocija: Za promocijo aplikacije lahko delimo spletni naslov (ang. link) na vseh družabnih omrežjih, kjer veliko lažje prepričamo uporabnika, ki se še ni srečal z aplikacijo, da odpre spletni naslov, kot da si aplikacijo prenese.

Slabosti spletnih aplikacij:

- Odzivnost: Ker mora aplikacija stalno komunicirati z aplikacijskim strežnikom, je odzivnost občutno nižja.

- Grafična podoba in izdelava aplikacije: V navadnih aplikacijah lahko veliko hitreje in lepše naredimo orodje, ki nam bo omogočalo premikanje in povezovanje objektov, saj za slednje obstaja veliko pripomočkov, ki jih za spletne aplikacije ne moremo uporabiti.
- Nujna povezava do spleta

Za razvoj spletne aplikacije sem si izbral ASP.NET MVC okolje zaradi občutka, da mi ponuja najbolj objektno usmerjeno rešitev in s tem tudi hitrejšo izdelavo. Za shranjevanje podatkov pa sem si izbral relacijsko podatkovno bazo na Microsoftovem SQL strežniku. Zaradi izbire MVC tehnologije sem ločil uporabniški vmesnik, poslovno logiko in podatkovni model, kot je to prikazano na Sliki 3.1.



Slika 3.1: Delovanje MVC-ja [18].

3.2 Podatkovni model uporabljene podatkovne baze

V tem razdelku je opisan logičen podatkovni model, katerega uporablja nastala spletna aplikacija. Vse podatkovne tabele in relacije med njimi so pri-

kazane na Sliki 3.2. V nadaljevanju so predstavljene vse tabele v logičnem podatkovnem modelu.

UserAccount

Tabela UserAccount je namenjena registraciji in vpisovanju v spletno aplikacijo. Aplikacija ob registraciji shrani v tabelo uporabniško ime, geslo in spletni naslov. Vpisano uporabniško ime in geslo se nato uporabi ob prijavi.

UserProject

Da sem uporabniku omogočil načrtovanje več projektov naenkrat, sem naredil tabelo UsersProject. Tabela hrani podatke o imenu, opisu in o nivoju načrtovanja. Za povezavo na uporabniški račun pa uporabim tuji ključ AccountId.

ModelData

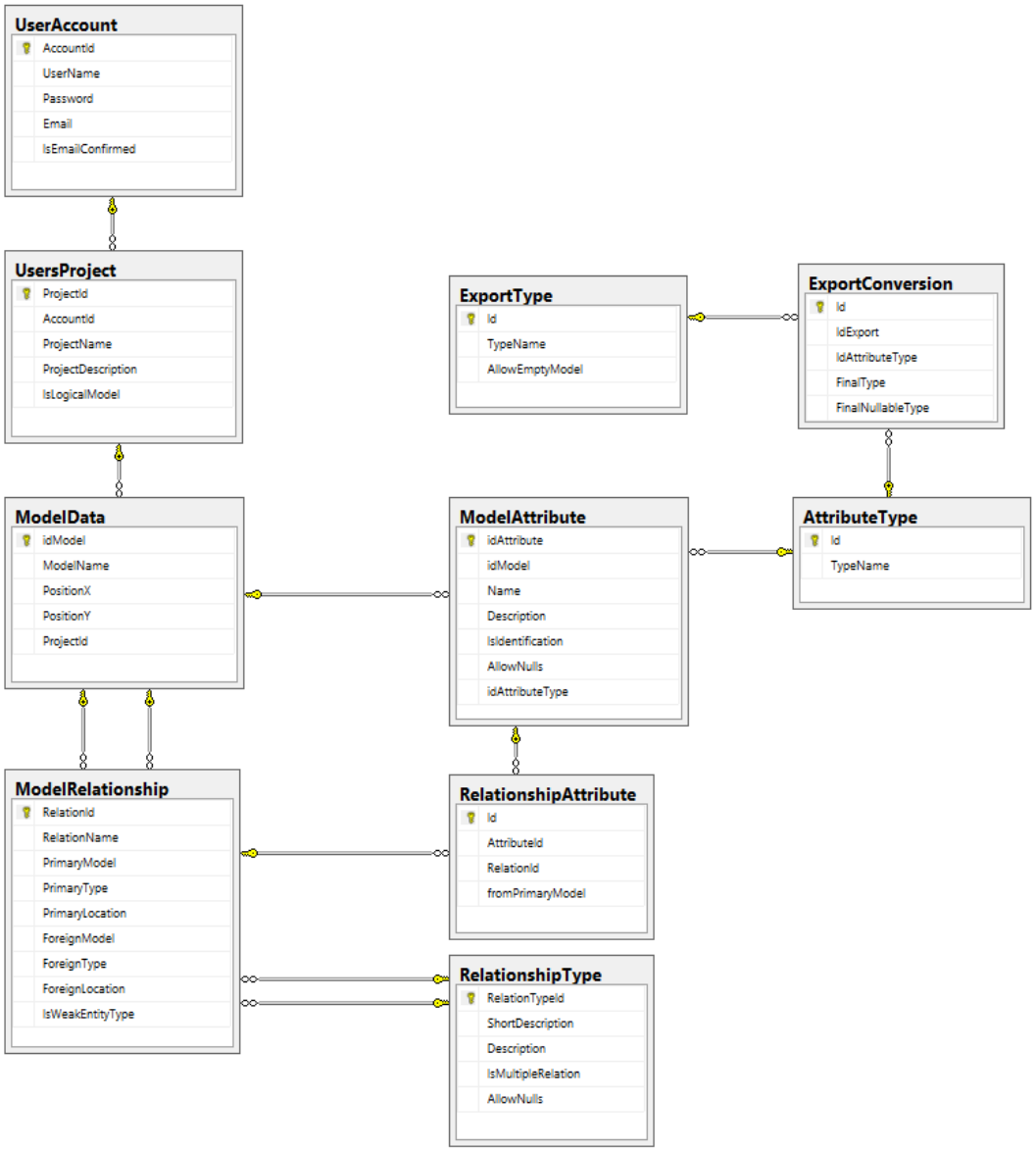
V vsakem projektu imamo lahko več entitetnih tipov, kasneje relacij. V tabeli poleg imena entitetnega tipa/relacije hranim tudi lego objekta na zaslonu.

ModelAttirubte

Tabela ModelAttribute je namenjena shranjevanju podatkov o atributih v entitetnemu tipu. Atributi vsebujejo informacijo o imenu, opisu, podatkovnem tipu, obveznosti in informacijo, ali je atribut identifikator entitetnega tipa.

AttributeType

V tabeli so shranjeni vsi podatkovni tipi, ki jih lahko uporabimo za tip posameznega atributa.



Slika 3.2: Diagram vseh tabel in relacij v podatkovni bazi.

ModelRelationship

Tabelo ModelRelationship uporabim za povezovanje entitetnih tipov oziroma za shranjevanje povezav. Zapisi v ModelRelationship tabeli vsebujejo informacijo o obeh entitetnih tipih in razmerju med njima.

RelationshipType

Vse številnosti, ki jih lahko uporabimo v povezavi, so opisane v tabeli RelationshipType.

RelationshipAttribute

Tabelo uporabim šele pri logičnem načrtovanju. Uporabim jo za označitev atributov, ki so uporabljeni v povezavi. (V konceptualnem nivoju te še ne potrebujem, ker v povezavah uporabljam samo identifikatorje entitetnih tipov).

ExportType

V tabeli ExportType so zapisani vsi programski jeziki in tipi podatkovnih baz, za katere imam implementirano pretvorbo v fizični model.

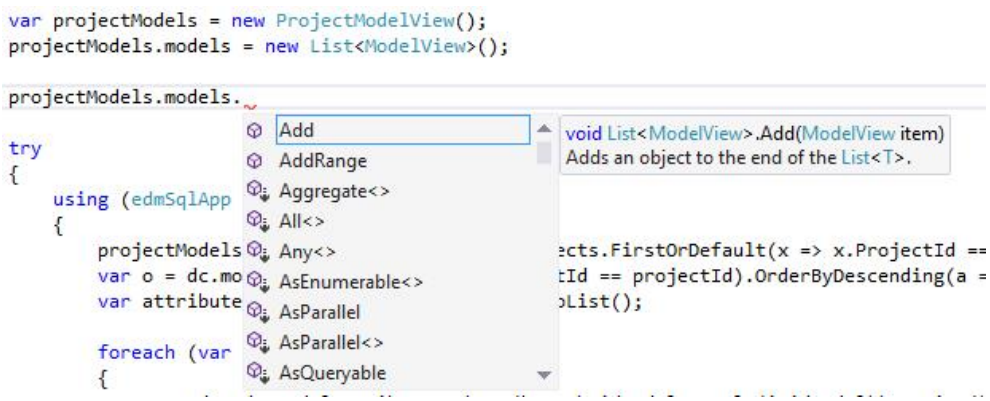
ExportConversion

Ker programski jeziki in podatkovne baze uporabljajo različne podatkovne tipe in njihove različne označbe, sem naredil tabelo ExportConversion. Tabela za vsak podatkovni tip iz tabele AttributeType pove, v kaj se pretvori ob pretvorbi v fizični model.

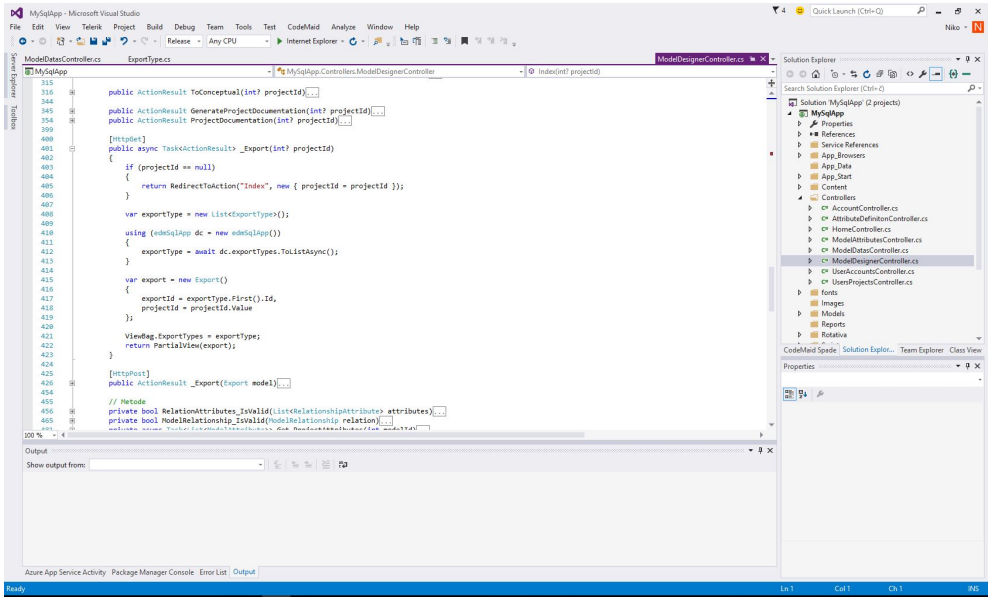
3.3 Uporabljena orodja in programski jeziki

3.3.1 Visual studio 2015

Visual studio je Microsoftovo razvojno okolje (gl. Slika 3.4), ki podpira pisanje programske kode v različnih programskih jezikih. Vsebuje urejevalnik kode, ki nam sproti zaznava in dokončuje kodo (ang. IntelliSense), hkrati pa tudi preverja samo pravilnosti sintakse (gl. Slika 3.3). V razvoju uporabniškega vmesnika mi je najbolj pomagal zaradi napredne funkcije za razhroščevanje (ang. debugging).



Slika 3.3: IntelliSense v Visual Studio 2015.

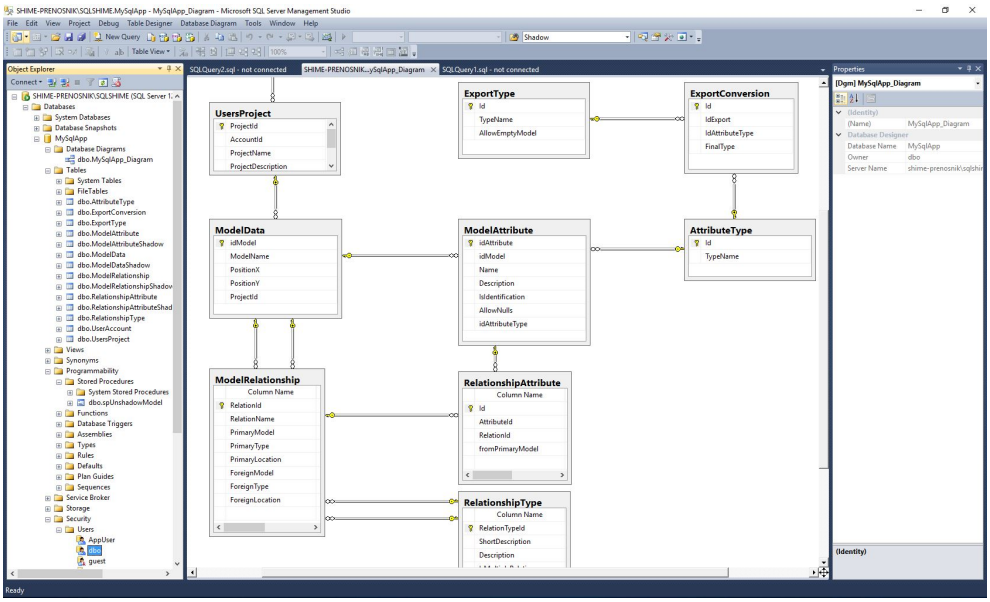


Slika 3.4: Orodje Visual Studio 2015.

3.3.2 Microsoft SQL Server Management Studio

SQL Server Management Studio (SSMS) je okolje za dostop, konfiguracijo, upravljanje, vodenje in razvoj vseh komponent SQL serverja [21].

Z orodjem, ki je prikazano na Sliki 3.6, sem načrtoval in izdelal vse podatkovne tabele, njihove atribute in medsebojne povezave.



Slika 3.5: Orodje Microsoft SQL Server Management Studio.

3.3.3 C Sharp

Programski jezik C Sharp je objektno usmerjen programski jezik, ki ga je razvil Microsoft [8]. Na videz je zelo podoben programskem jeziku Java. V izdelku je uporabljen predvsem za manipulacijo s podatki.

3.3.4 Linq

Language-Integrated Query (LINQ) je eden od elementov, ki je bil predstavljen ob izidu Visual Studia 2008 [14]. Linq se od ostalih poizvedovalnih jezikov razlikuje po tem, da omogoča poizvedbe nad podatki z uporabo operatorjev uporabljenega programskega jezika (gl. Slika 3.6).

3.3.5 Entity framework

Entity framework nam omogoča pretvorbo zapisov iz podatkovne baze v objekte/razrede v aplikaciji. Obnaša se kot nekakšen preslikovalec (ang. mapper). Ker skrbi za komunikacijo med podatkovno bazo in podatkovnim

```
var relations = new List<ModelRelationship>();

relations = await (from models in dc.modelDatas
                   join relation in dc.modelRelationships on models.idModel equals relation.PrimaryModel
                   where models.ProjectId == projectId
                   select relation).ToListAsync();
```

Slika 3.6: Primer uporabe LINQ, ki za dostop do podatkovne baze uporablja Entity framework.

modelom v aplikaciji (razredi), se lahko lažje osredotočimo na manipulacijo s podatki.

Entity framework omogoča dva različna pristopa za generiranje podatkovnega modela (naših razredov). V okviru prvega pristopa Entity framework generira podatkovni model na podlagi obstoječe podatkovne baze, kar omogoča hitrejši začetek uporabe podatkovnega modela. Slaba lastnost opisanega tipa se pojavi ob nadgradnji ali spremembi podatkovne baze, ker prepiše obstoječe razrede z na novo generiranimi in tako izgubimo vse lastnosti razredov, ki smo jih dodali ob izdelavi aplikacije. V okviru drugega pristopa je zgodba obrnjena in mi skrbimo za kreiranje podatkovnega modela (razredov), Entity Framework pa pred vsakim zagonom aplikacije preveri, če tabele v podatkovni bazi ustrezajo lastnostnim razredov. Če tabele ne obstajajo ali pa ne ustrezajo lastnostim razredov, jih Entity Framework sam ustvari/spremeni.

3.3.6 HTML

Hyper Text Markup Language (HTML) je označevalni jezik, ki se ga uporablja za oblikovanje spletnih strani [13]. S slednjim sem dodal na spletno stran vse naše kontrole (npr. vnosna okna, menije, tabelice itd.).

3.3.7 CSS

Cascading Style Sheets (CSS) je oblikovni jezik, ki je namenjen oblikovanju spletnih strani. Z njim lahko enostavno spreminjamo barve in oblike vsem

HTML elementom.

3.3.8 jQuery

JQuery je obogatena JavaScript knjižnica. Omogočila mi je iskanje, manipuliranje in validacijo nad vsemi HTML elementi.

3.3.9 jsPlumb

JsPlumb je JavaScript knjižnica, ki za svoje delovanje potrebuje JQuery knjižnico. Slednja knjižnica je namenjena predvsem za prikaz in risanje HTML elementov. Uporabil sem jo za risanje povezav med entitetnimi tipi. Na Sliki 3.7 je prikazana nastavitev oblike povezave med dvema entitetnima tipoma.

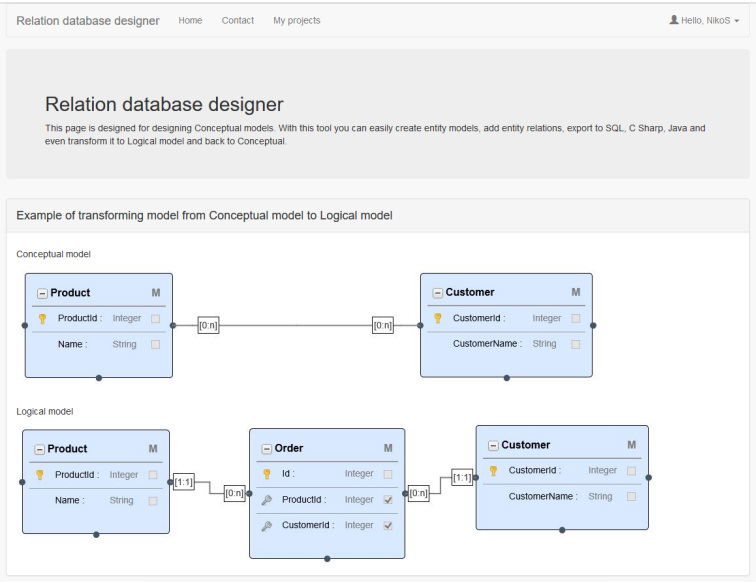
```
function GetEndPointOption(position) {

    // ["Right", "Left", "Top", "Bottom"]

    // Tests for lines.
    var endpointOptions = {
        anchor: position,
        isSource: true,
        isTarget: true,
        endpoint: ["Dot", { radius: 5 }],
        connector: "Flowchart",
        connectorStyle: { lineWidth: 2, strokeStyle: 'gray' },
        maxConnections: -1,
        dragAllowedWhenFull: true
    };

    return endpointOptions;
}
```

Slika 3.7: Primer nastavljanja oblike povezave s knjižnico jsPlumb.

Poglavje 4

Predstavitev izdelane aplikacije

V tem poglavju bom predstavil spletno aplikacijo, ki je bila izdelana v okviru diplomskega dela. Aplikacija, ki sem jo poimenoval Relation database designer, uporabniku ponuja možnost načrtovanja relacijske podatkovne baze. Sestavljena je iz glavnega okna, v katerem je prikazana vsebina, in menija na vrhu, ki je statičen oziroma enak na vseh podstraneh (gl. Slika 4.1).

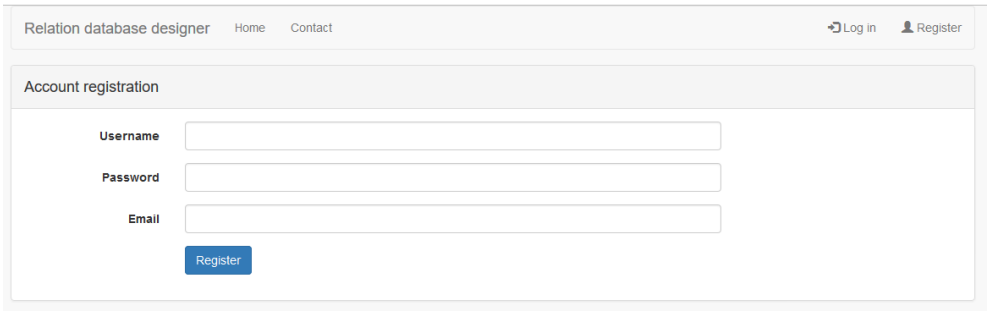


Slika 4.1: Vstopna stran aplikacije.

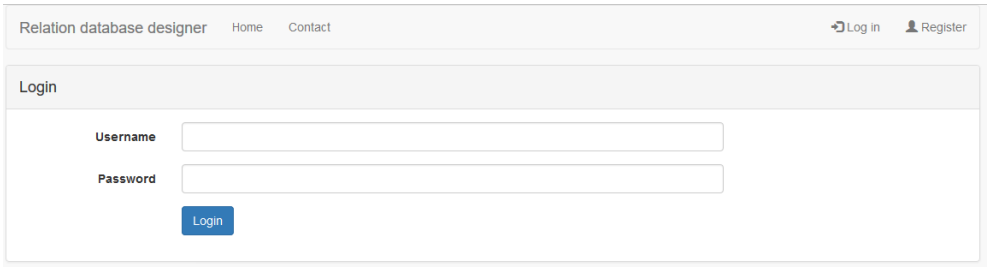
4.1 Registracija uporabnika

Ker imamo možnost kreiranja več projektov, projekti pa so vezani na vsakega posameznega uporabnika, se moramo pred vstopom v aplikacijo registrirati. Gumbi, ki nudijo možnost registracije, vpisa in urejanja uporabnikovih podatkov se v uporabniškem vmesniku nahajajo desno zgoraj, kot je vidno na sliki 4.2.

Za registracijo sem uporabil samo najosnovnejše podatke, kot so uporabniško ime, geslo in e-poštni račun. Registracija je uspešna, ko izpolnimo vse tri podatke in pritisnemo na gumb 'Register'. Ob registraciji se podatki shranijo v podatkovno bazo.



Slika 4.2: Registriranje uporabnika.



Slika 4.3: Vpis v aplikacijo.

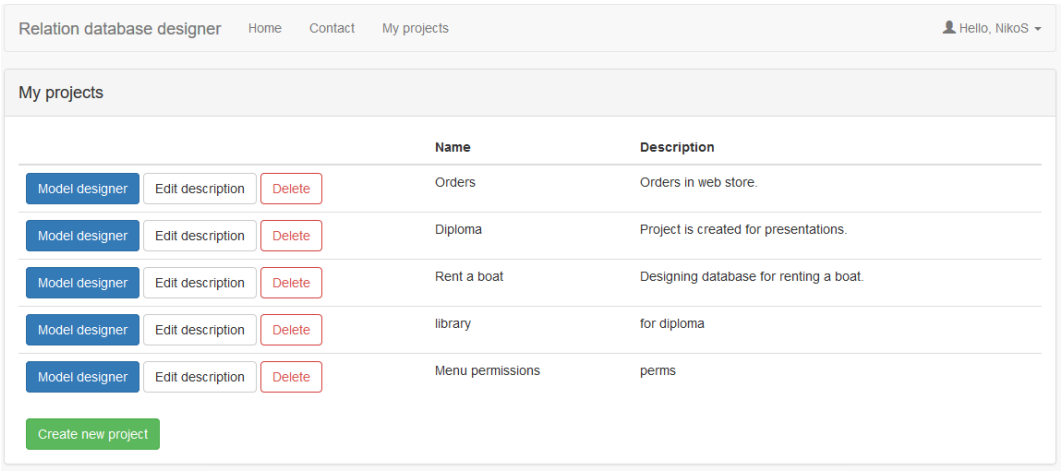
Ko je uporabnik registriran, se naslednjič samo prijavi. Prijavimo se lahko tako, da v zgornjem meniju kliknemo na gumb z imenom "Log in", vpišemo

naše uporabniško ime in geslo in nato kliknemo moder gumb "Login"(gl. Slika 4.3).

Tako po registraciji, kot tudi po prijavi v aplikacijo, se uporabnikovi podatki shranijo v piškotek (ang. cookie). Piškotek služi za prikaz in urejanje uporabnikovih podatkov in njegovih projektov. Uporabnik ga lahko izbriše tako, da zapre brskalnik, ali pa se samo enostavno odjavi.

4.2 Kreiranje in ločevanje projektov

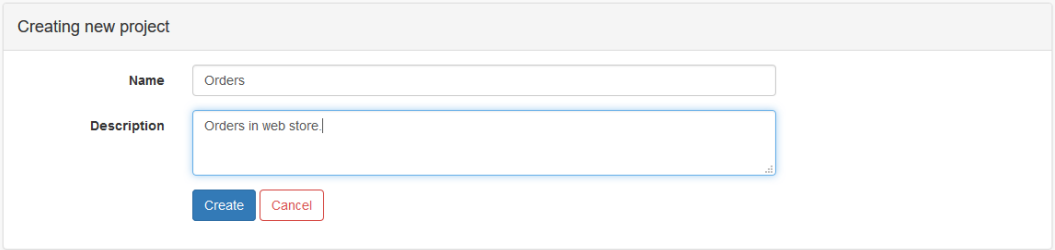
Razvoj podatkovne baze lahko traja od nekaj minut pa do nekaj mesecev, med tem časom pa lahko načrtovalec dela na več različnih projektih. Da uporabnik v aplikaciji ne bi bil omejen samo na en razvoj podatkovne baze, sem omogočil možnost ločevanja na projekte.



Slika 4.4: Pregled uporabnikovih projektov.

Pregled in vnos sta mogoča ob kliku na gumb 'My projects', ki se nahaja v meniju zgoraj. S klikom na gumb se nam odpre nova stran (gl. Slika 4.4), ki je sestavljena iz pregledovalne tabele in gumba za vnos novega zapisa, ki se nahaja pod njo. V pregledovalni tabeli imamo seznam naših projektov, ki je sestavljen iz imena, opisa in ukaznih gumbov. Z gumboma 'Delete' in 'Edit

description' lahko brišemo in urejamo zapise, z gumbom 'Model designer' pa se nam odpre nova stran, kjer lahko pregledujemo in urejamo entitetne tipe. Na Sliki 4.6 je prikazana vnosna maska za vnos novega projekta.



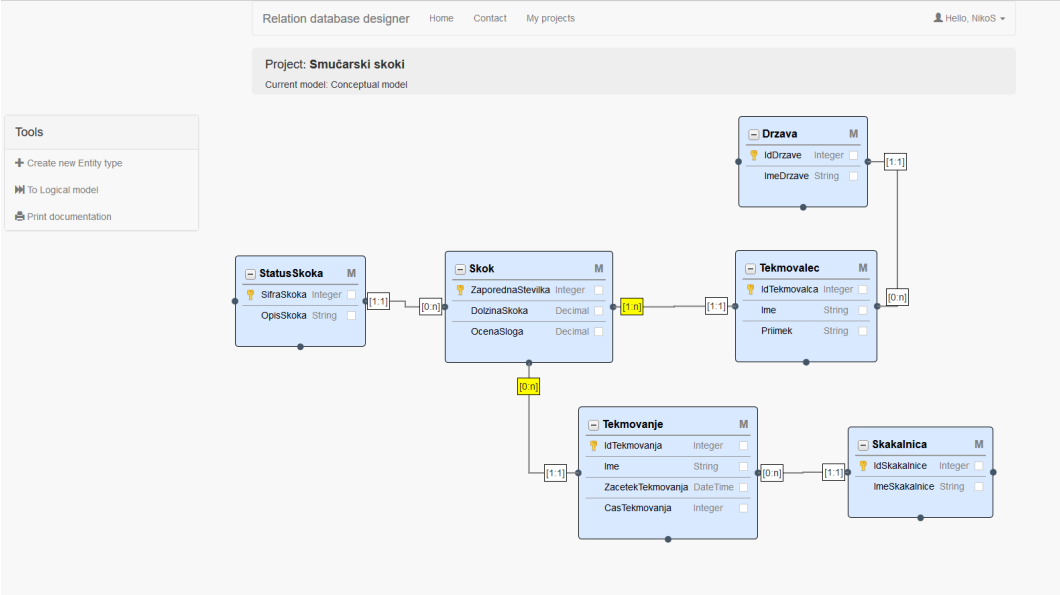
Slika 4.5: Primer kreiranja novega projekta.

4.3 Načrtovanje konceptualnega in logičnega modela

Vnosna maska za načrtovanje konceptualnega modela (gl. Slika 4.6) je sestavljena iz treh segmentov. V prvem segmentu, ki je pod glavnim menijem, imamo napisano ime projekta in modela, ki ga trenutno načrtujemo. V segmentu, ki se nahaja skrajno levo, imamo vsa orodja, ki jih lahko uporabimo v danem trenutku. Zadnji segment pa je delovna površina, na kateri so prikazani vsi entitetni tipi in njihove relacije.

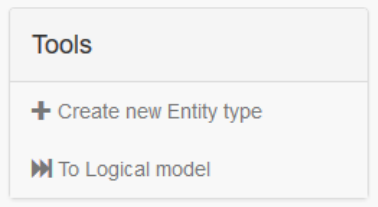
4.4 Orodja

Ob kreiranju novega projekta je delovna površina Model designer-ja čisto prazna in zato imamo na voljo manj možnih orodij. Dokler ne naredimo vsaj enega entitetnega tipa, sta na vnosni maski vidna samo dva gumba, gumb za kreiranje novega entitetnega tipa 'Create new Entity type' in gumb za pretvorbo v logični model 'To Logical model', kot je vidno na Sliki 4.7. Kasneje, ko naredimo vsaj en entitetni tip, se nam prikaže gumb, s katerim



Slika 4.6: Vnosna maska za prikaz in izdelavo konceptualnega in logičnega modela.

lahko natisnemo dokumentacijo, in ko celotni model pretvorimo v logični model, se nam prikažeta še dva gumba. Eden nam omogoča pretvorbo nazaj v konceptualni model, drugi pa v fizični model (na vnosni maski je poimenovan 'Export').



Slika 4.7: Okno, v katerem so prikazana vsa orodja.

4.4.1 Opis gumbov/orodij

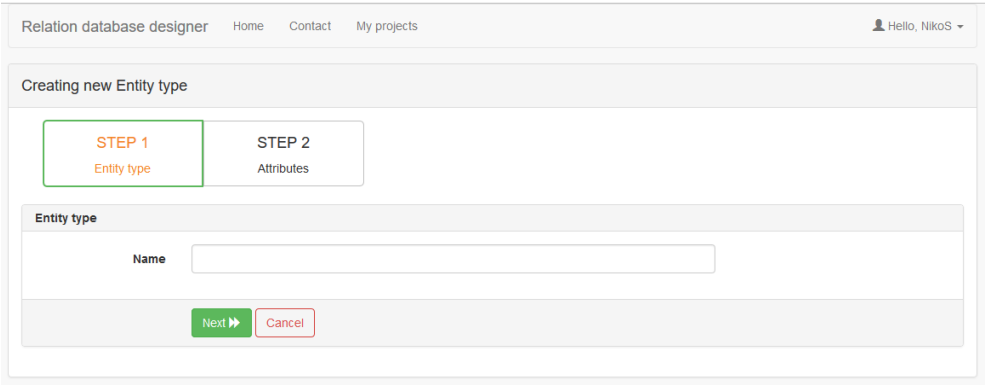
- gumb 'Create new Entity type': gumb odpre vnosno masko za dodajanje entitetnega tipa in njegovih atributov;
- gumb 'Create new Relation': nam omogoča dodajanje novih relacij (tabel) v logičnem modelu;
- gumb 'To Logical model': konceptualni model pretvori v logični model;
- gumb 'To Conceptual model': logični model pretvori nazaj v konceptualni model;
- gumb 'Print documentation': v novem oknu odpre PDF dokument, ki vsebuje opis projekta, entitetnih tipov in njegovih atributov;
- gumb 'Export': nam ponudi možnost pretvorbe logičnega modela v SQL (za SQL server ali DB2 podatkovno bazo), C sharp, Javo in Visual basic.

4.5 Dodajanje in urejanje novih entitetnih tipov

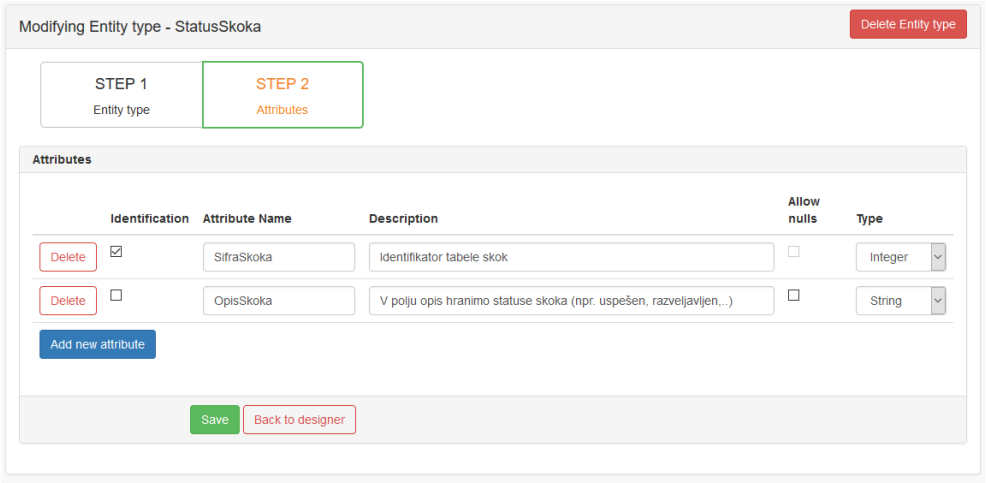
Da bi uporabnik lažje in predvsem na bolj razumljiv način vnesel nov entitetni tip, sem vnosno masko razdelil na dva koraka (gl. Slika 4.8). Z ločenima korakoma sem naredil delovni tok, kjer v prvem koraku vneseš ime entitetnega tipa, v drugem pa vse atribute in njihove lastnosti (gl. Slika 4.9).

4.6 Prikaz entitetnih tipov in dodajanje relacij

Ko zaključimo z vnosom entitetnih tipov in atributov, je vsak posamičen entitetni tip viden kot moder pravokotnik (gl. Slika 4.10). Vsak entitetni

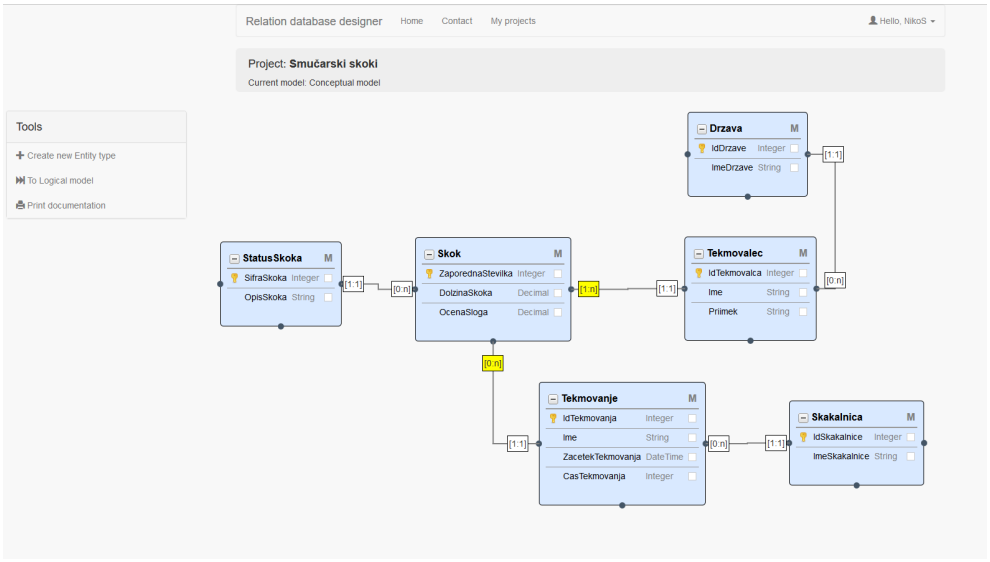


Slika 4.8: Vnos novega entitetnega tipa.

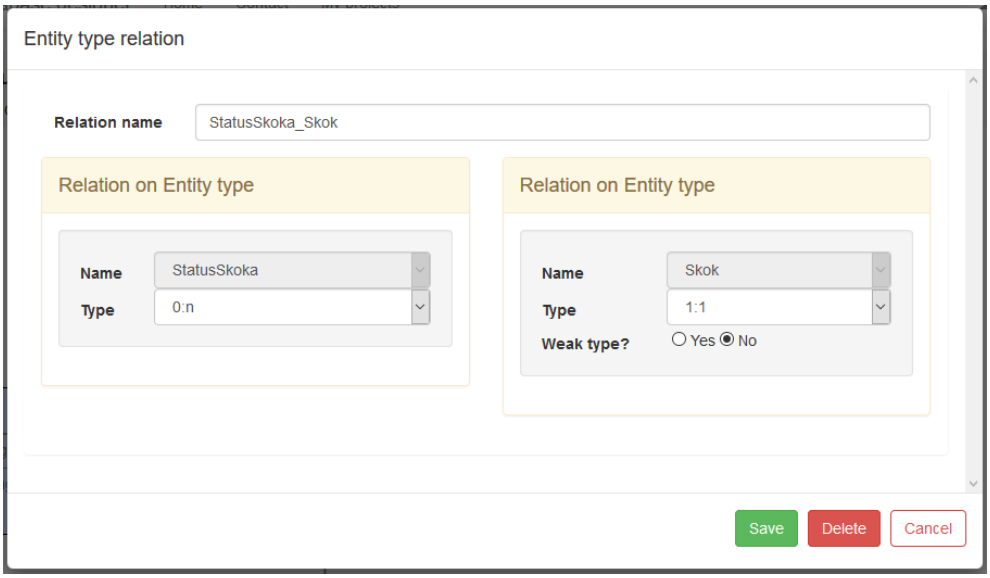


Slika 4.9: Dodajanje atributov v entiteto.

tip lahko 'primemo' in ga poljubno premikamo po ekranu, ali pa ga s klikom na gumb, ki vsebuje minus, zmanjšamo ali povečamo. Na vsakem robu pravokotnika je vidna siva pika, ki je namenjena povezovanju entitetnih tipov. Povežemo jih tako, da pritisnemo na piko ob prvem entitetnem tipu in jo vlečemo do pike entitetnega tipa, ki ga želimo povezati. Ob povezavi dveh entitetnih tipov se odpre pojavno okno (gl. Slika 4.11), na katerem določimo, katera entitetna tipa želimo povezati in kakšno želimo razmerje med njima.



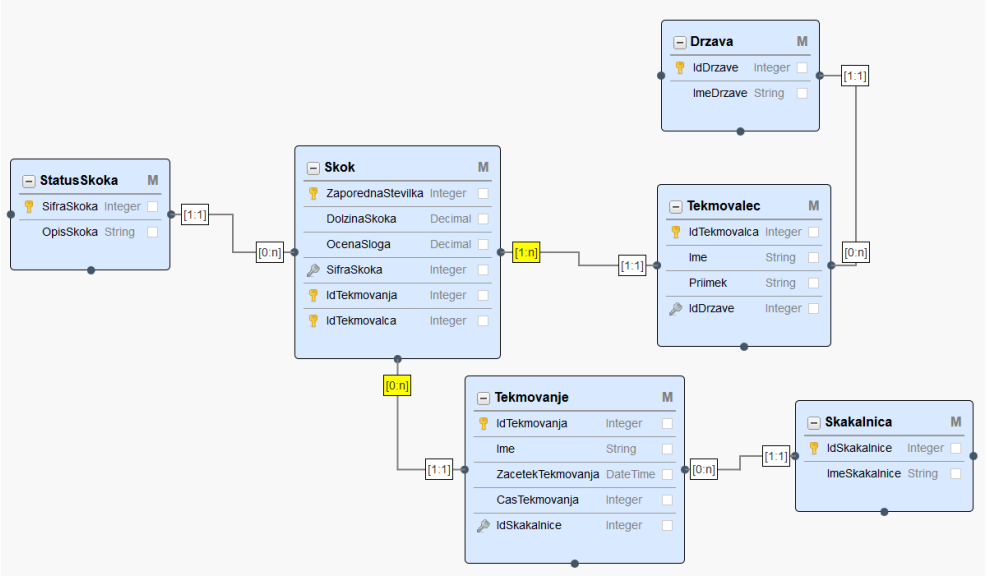
Slika 4.10: Prikaz vseh dodanih entitetnih tipov in relacije med njimi.



Slika 4.11: Primer dodajanja relacije.

4.7 Pretvorba v logični model in fizični model

Ko imamo konceptualni model izdelan, lahko s klikom na gumb 'To logical model' pretvorimo konceptualni model v logičnega. Ob pretvorbi aplikacija z upoštevanjem vseh pravil za pretvorbo doda vse tuje ključe in se znebi 'm:n' povezav. Primer pretvorbe je viden na Sliki 4.12.



Slika 4.12: Pretvorba konceptualnega modela iz Slike 4.11 v logični model.

Na koncu lahko logičen model z gumbom 'Export' izvozimo v tekstovni dokument, ki predstavlja fizični model (gl. Slika 4.13). V datoteki se nahaja koda za kreiranje tabel v podatkovni bazi oziroma nam v primeru izvoza v programski jezik vrne narejene razrede. Možnost izvoza v programske jezike je bila dodana, da si lahko naredimo približek razrednega diagrama in da bi načrtovalcem programske opreme poleg pomoči pri načrtovanju in izdelavi podatkovne baze, pomagali še pri izdelavi podatkovnega modela v aplikaciji.

Slika 4.13: Pojavno okno, s katerim lahko naredimo fizični model.

Poglavje 5

Sklepne ugotovitve

V diplomskem delu sem razvil spletno aplikacijo, ki zajema celotno načrtovanje relacijske podatkovne baze. Pred razvojem sem bil zelo negotov o izdelku, ker se še nikoli nisem srečal niti s programskim ogrodjem ASP.NET MVC, niti z risanjem premikajočih elementov, niti z risanjem črt, ki bi jih lahko uporabil za povezave. Na koncu sem poleg narejenega izdelka, ki deluje v vseh posodobljenih brskalnikih (pogoj je podpiranje oblikovnega jezika HTML5 in programskega jezika JavaScript), pridobil tudi zelo veliko novega znanja.

Za zaključek sem spletno aplikacijo postavil v oblak (Azure), kjer sem moral narediti nov podatkovni in aplikacijski strežnik. Pri vzpostavitvi SQL strežnika v oblaku sem imel malo več ročnega dela, ker sem imel na voljo samo ukazno vrstico, s pomočjo katere sem moral napisati vse ukaze za kreiranje tabel in ostalih objektov v bazi, ki jo uporablja razvita aplikacija, nato pa še za napolnitev nekaterih podatkov. Vzpostavitev aplikacijskega strežnika in aplikacije v oblaku pa je bila končana z nekaj kliki.

Kot vsaka aplikacija ima tudi ta določene pomanjkljivosti, ki se jih lahko v nadaljevanju odpravi. Ena izmed pomanjkljivosti je pomanjkanje navodil in pomoči. Poleg pomanjkljivosti ima sistem veliko potencialnih nadgradenj. Ena izmed zelo dobrodošlih nadgradenj bi lahko bila dodati možnost pretvorbe v fizični model še za katero drugo pogosto uporabljeno podatkovno bazo. Naslednja zelo dobrodošla nadgradnja bi bila možnost odpiranja apli-

kacije v mobilnih brskalnikih, saj bi na tak način lahko načrtovali podatkovno bazo tudi v stanju mobilnosti.

Literatura

[1] Korth F. Henry, Silberschatz Abraham. *Database System Concepts*. New York: McGraw-Hill, 1991.

[2] Rob Peter, Coronel Carlos. *Database systems : design, implementation, and management*. Boston: Course technology, 2009.

[3] Thomas M. Connolly, Carolyn E. Begg. *Database Systems, A Practical Approach to Design, Implementation and Management, Fourth Edition*. Addison-Wesley, 2005.

[4] Toby Teorey, Sam Lightstone, Tom Nadeau. *Database Modeling and Design: Logical Design*. Elsevier, 2006.

[5] Tomaž Mohorič. *Načrtovanje relacijskih podatkovnih baz*. Založba BiTIM 1997.

Spletni viri

[6] Conceptual data modeling. [Online]. Dosegljivo: http://www.ariscommunity.com/users/eva-klein/2012-12-28-conceptual-data-modeling-aris-using-er-models-motivation. [Dostopano 24. 2. 2016].

[7] Conceptual models. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Conceptual_model/. [Dostopano 24. 2. 2016].

[8] C Shar (programming language). [Online]. Dosegljivo: https://en.wikipedia.org/. [Dostopano 24. 2. 2016].

[9] Data modeling. [Online]. Dosegljivo: http://www.1keydata.com/datawarehousing/data-modeling-levels.html. [Dostopano 24. 2. 2016].

[10] Data model symbols. [Online]. Dosegljivo: http://www.agiledata.org/images/dm101NotationSummary.gif. [Dostopano 24. 2. 2016].

[11] Definition of Modeling the Data Domain. [Online]. Dosegljivo: https://www.chegg.com/homework-help/definitions/modeling-the-data-domain-3. [Dostopano 24. 2. 2016].

[12] GNU General Public Licence. [Online]. Dosegljivo: https://www.gnu.org/copyleft/gpl.html. [Dostopano 20. 9. 2014].

[13] HTML5. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/HTML5. [Dostopano 24. 2. 2016].

[14] Introduction to LINQ. [Online]. Dosegljivo: https://msdn.microsoft.com/en-us/library/bb397897.aspx. [Dostopano 24. 2. 2016].

[15] jsPlumb. [Online]. Dosegljivo: https://jsplumbtoolkit.com/docs/toolkit/home.html. [Dostopano 24. 2. 2016].

[16] Podatkovna normalizacija. [Online]. Dosegljivo: http://drenovec.tsckr.si/model/normal.htm. [Dostopano 24. 2. 2016].

[17] Slika agilne metodologije. [Online]. Dosegljivo: http://www.saop.si/images/custom/Poslovne informacije/taktika/saop-agilno.vodenje.jpg. [Dostopano 15. 9. 2016].

[18] Slika delovanja MVC-ja. [Online]. Dosegljivo: http://book.cakephp.org/2.0/en/cakephp-overview/understanding-model-view-controller.html. [Dostopano 24. 2. 2016].

[19] Sql - indexes. [Online]. Dosegljivo: http://www.tutorialspoint.com/sql/sql-indexes.htm. [Dostopano 4. 10. 2016].

[20] Tehnike načrtovanja konceptualnega modela. [Online]. Dosegljivo: http://www.s-sers.mb.edus.si/gradiva/rac/moduli/podatkovne_baze/11_podatkovna_pla
[Dostopano 3. 10. 2016].

[21] Uporaba orodja SQL Server Management Studio. [Online]. Dosegljivo: https://msdn.microsoft.com/en-us/library/ms174173.aspx. [Dostopano 19. 9. 2014].